RESEARCH PAPER

SWIRRL. Managing Provenance-aware and Reproducible Workspaces

Alessandro Spinuso[1†], Mats Veldhuizen[1], Daniele Bailo[2], Valerio Vinciarelli[2] & Tor Langeland[3]

[1]Koninklijk Nederlands Meteorologisch Instituut, De Bilt, Utrecht 3731 GA, The Netherlands

[2]Istituto Nazionale Geofisica e Vulcanologia, Rome, Lazio 00143, Italy

[3]NORCE Norwegian Research Centre AS, Bergen, Hordaland NO-5838, Norway

Keywords: Productivity; Data-analysis; Reproducibility; Provenance; Cloud-computing

Citation: Spinuso, A., et al.: SWIRRL. Managing provenance-aware and reproducible workspaces. Data Intelligence 4(2), 243-258 (2022). doi: 10.1162/dint_a_00129

Received: July 30, 2021; Revised: November 24, 2021; Accepted: February 5, 2022

ABSTRACT

Modern interactive tools for data analysis and visualisation are designed to expose their functionalities as a service through the Web. We present in this paper a Web API (SWIRRL) that allows Virtual Research Environments (VREs) to easily integrate such tools in their websites and re-purpose them to their users. The API deals, on behalf of the clients, with the underlying complexity of allocating and managing resources within a target cloud platform. By combining storage and containerised services, offering analysis notebooks and other visualisation software, the API creates dedicated working sessions on-demand, which can be accessed collaboratively. Thanks to the API's support for workflow execution, SWIRRL workspaces can be automatically populated with data of interest collected from external data providers. The system keeps track of updates and changes affecting the data and the tools by adopting versioning and standard provenance technologies. Users are provided with interactive controls enabling traceabilityand recovery actions, including the possibility of creating executable snapshots of their environments. SWIRRL is built in cooperation with two research infrastructures in the field of solid earth science and climate data modeling. We report on the particular adoptions and use cases.

[†] Corresponding author: Alessandro Spinuso (Email: alessandro.spinuso@knmi.nl; ORCID: 0000-0002-0077-8491).

1. INTRODUCTION

The combination of cloud platforms and containerised software has become a de-facto standard when deploying and managing remote computational services. However, although containerisation has the advantage of being portable across infrastructures, it requires specific expertise and special attention when reproducibility has to be achieved. This is particularly relevant in modern computational research, where not only the data and the analysis methods are frequently updated, but also the computational environment where the analysis takes place. In such a context humans make choices and interact with sophisticated software agents. This makes the environment evolve, suggesting the need to record the effects of the changes in order to restore toa past state, if needed, as well as allowing progress to be shared among peers. Technical formats, such as containers descriptors (e.g., Dockerfiles[1]) and environment files can be used to implement such use cases. However, they typically overlook the data and require technical knowledge to be used for re-enactment. This motivated the implementation of SWIRRL, a Web API that allows the automated deployment of computational workspaces. The API's high-level functions hide the complexity of managing the underlying cloud infrastructure, empowering clients to control the execution of workflows (e.g., for data staging) and to combine computational and software management services. These consists of processing and visual analytic tools that are re-purposed to the users on a determinate but yet extensible collection of data.

With the invocation of the API's methods, SWIRRL generates provenance information to describe the relationships between the different resources involved in the progress of the analysis. Provenance is used to assists users in obtaining accurate reproducibility, besides delivering actionable and interoperable documentation. Ultimately, our conceptual design and technical solution, approach challenges that are of interest for the CWFR[2]. We target the realisation of an infrastructure service that supports literate programming [8] and visual analytic in combination with workflows, manages the cloud and fosters FAIRness by design.

2. RELATED WORK

Achieving reproducible science is typically addressed at the level of the particular tool that offers computational capabilities to data analysts and developers. Tools may consist of domain specific software libraries, [4], as well as general purpose workflow management systems [2, 15, 6]. Recently, the interactive environment offered by Jupyter Notebook[3] has gained attention because of its advanced literate programming capabilities [14]. Although this shows the potential for combining exploration and explanation of new analysis methods in a common framework, Rule *et al*. explain how reproducibility is better achieved when users develop their notebooks following general purpose guidelines [13]. Eventually, to really reproduce the analysis, users have be familiar with additional systems, (i.e., data and software repositories) [1], requiring them to learn new technologies and formats. Obtaining actionable metadata that would also foster the automation of

[1] https://docs.docker.com/engine/reference/builder/
[2] https://fairdo.org/wg/fdo-cwfr/
[3] https://jupyter.org

reproducibility via generic systems and protocols, is one of the core aspects of the FDO [3] architecture. These concepts are extremely relevant for the design and future refinements of our Web API, which aims at empowering Virtual Research Environments (VREs) with reproducible interactive services and traceable data management operations. SWIRRL adopts the most recent technologies in the field of microservices and provenance management. Its novelty, compared to systems such as JupyterHub[④] and Everware [16], is the adoption of provenance information to drive the automated enactment of traceability and reproducibility scenarios, besides its data-driven deployment model, which fosters collaborative interactions in focused workspaces. Reproducibility is not limited to notebooks, but includes data and workflows. We are currently working to extend this capability to other services for interactive visual analysis [9].

3. SYSTEM OVERVIEW

SWIRRL is accessible as a Web API. Our target clients are VREs, looking to extend their functionalities with computational services that can be easily integrated and that manage the underlying infrastructure autonomously. As a concrete use cases, we shall take into account existing community portals (i.e., EPOS Data Portal[⑤] and *Climate4Impact*[⑥]). Both offer scientists a discovery tool for data that is available through a variety of distributed data services. Once the users have identified the data of interest, the portals give the option to collect, access and manipulate the data onto a dedicated *Workspace*. At this stage, the SWIRRL API comes into play by offering the flexibility to orchestrate and control the workspaces in different ways. This is built around four core concepts: *Session*, *Workflow*, *Service* and *ServicePool* (Figure 1).

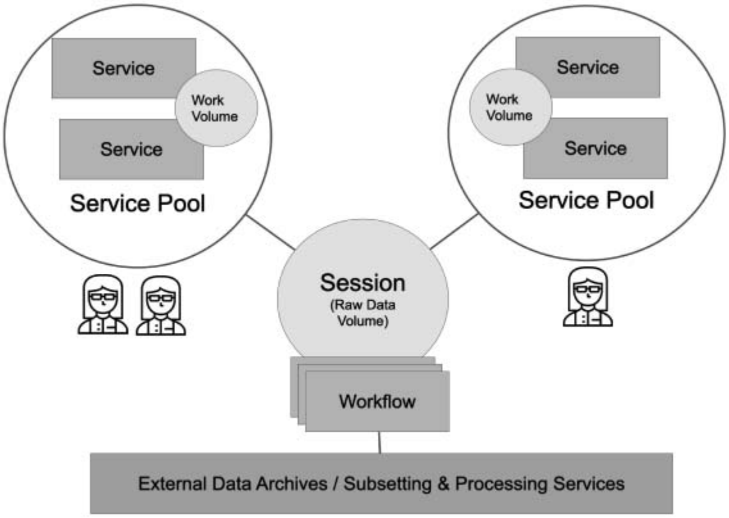


Figure 1. Orchestration of *Session*, *Workflow*, *Service* and *ServicePool* in SWIRRL workspaces.

[④] https://jupyter.org/hub
[⑤] https://www.ics-c.epos-eu.org/
[⑥] https://dev.climate4impact.eu

A *Session* consists of a read-only *Data Volume* which is populated and modifiable only via a *Workflow*. Here versioning of the data and provenance associated with the workflow is automatically generated and managed. A *ServicePool* is attached to a *Session*. It contains a read-write *Work Volume* and groups one or more instances of a *Service* that offer users specific analysis capabilities. All the services in the pool share the *Data Volume* and *Work volume*. In the latter, results of the analysis can be jointly manipulated. A *ServicePool* could be visible to one or more users, according to design choices of the particular VRE.

These design principles are translated into general parametrisation controls of the SWIRRLAPI, to facilitate clients in the orchestration of the workspaces. However, given the demand of the EPOS[7] and ENES[8] communities, the current implementation offers *JupyterLab*[9] and the *Enlighten Web* visualisation tool [9] as possible *Service* instances. The former has become a very popular tool among scientists in multiple domains, the latter is a visual analytic tool, which addresses the needs of the EPOS community. Depending on requirements, SWIRRL can be extended to accommodate additional tools. New API methods can be implemented by simply building on the same contracts and deployment mechanisms developed for the other services, with little adaptations to better deal with the characteristics of a particular analysis tool. In future, thanks to the contribution of the CWFR initiative, which identifies commonalities in combining, describing and reproducing operations performed during the research practice, we foresee improvements in the definition of the API to enable a more homogeneous and seamless control of different classes of services.

As shown in Figure 2, SWIRRL (which is available in open-source[10]) strongly relies on the containerised orchestration of the services it provides. It exploits infrastructures offering a Kubernetes[11] cluster to manage and distribute containerised applications across a set of computational nodes. Provenance is generated for every invocation of the methods of the API. This is achieved by adopting common provenance representations and technologies, such as the PROV-Template specification [12] and PROV-Template catalogue[12]. Once the provenance is generated, it gets stored in the *Neo4j*[13] graph database, as this has been already adopted by other provenance storage systems [11]. The database is exposed via the SWIRRL API with a collection of methods for provenance acquisition and interrogation. The current implementation supports two types of queries returning, respectively, (1) a (filtered) list of all the activities performed on a particular *Session* or *Service*, and (2) the provenance of an activity, given its *id*. The results are returned to clients in JSON-LD[14] format. In the following sections we will refer to such information showing how it is used to communicate changes to the users and support traceability, recovery and reproducibility actions.

[7] https://www.epos-eu.org

[8] https://is.enes.org

[9] https://jupyter.org/

[10] https://gitlab.com/KNMI-OSS/swirrl/swirrl-api

[11] https://kubernetes.io/

[12] https://github.com/EnvriPlus-PROV/ProvTemplateCatalog

[13] http://neo4j.org

[14] https://json-ld.org/

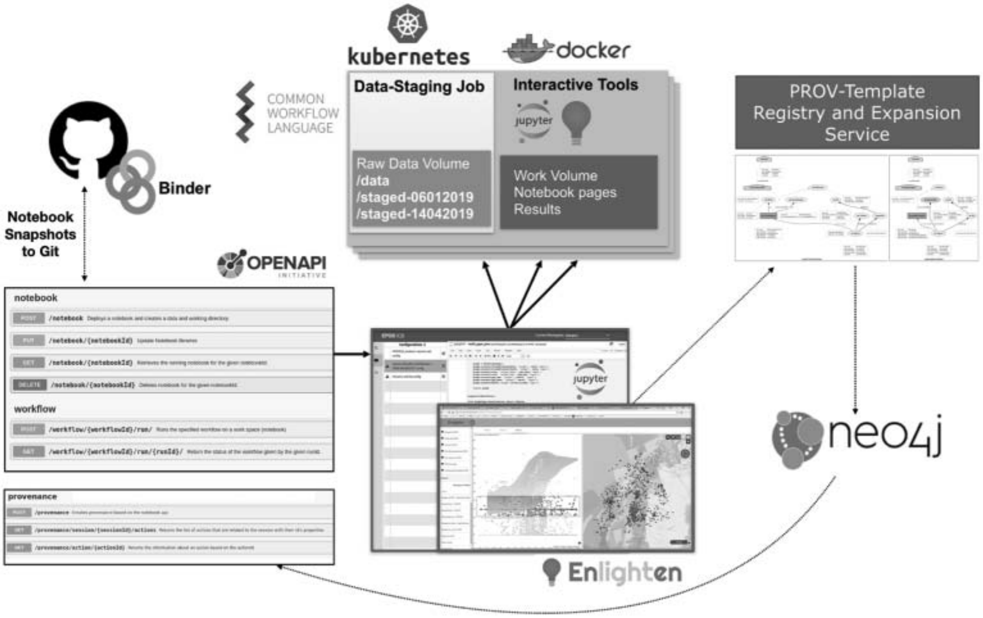

kubernetes
docker
Data-Staging Job
Interactive Tools
jupyter
Raw Data Volume
/data
/staged-06012019
/staged-14042019
Work Volume
Notebook pages
Results
PROV-Template Registry and Expansion Service
COMMON WORKFLOW LANGUAGE
Binder
Notebook Snapshots to Git
OPENAPI INITIATIVE
notebook
POST /notebook
PUT /notebook/{notebookId}
GET /notebook/{notebookId}
DELETE /notebook/{notebookId}
workflow
POST /workflow/{workflowId}/run/
GET /workflow/{workflowId}/run/{runId}/
provenance
POST /provenance
GET /provenance/session/{sessionId}/actions
GET /provenance/action/{actionId}
jupyter
neo4j
Enlighten

Figure 2. SWIRRL. Components overview.

4. DATA STAGING AND VERSIONING

A Research Infrastructure (RI) could host SWIRRL close to its data archive, minimising the overhead brought by transferring the data via the Internet. However, modern RIs often rely on a network of geographically distributed data services. This type of architecture, which is motivated by choices that are either technical or related to governance, challenges the efficient support of those analysis use cases that require access to data which is not co-located. For instance, the EPOS infrastructure is characterised by independent and multidisciplinary data providers, while ESGF[⑧] (accessible via *Climate4Impact*), disseminates the most updated and complete collection of climate models data via the distributed services of its federation. Acknowledging such scenarios motivated us to include within the API a workflow dedicated to the execution of data staging operations, from external providers to a target *Session*. The *Data Staging Workflow* takes in input a list of data URLs and triggers the download process. The files are saved and organised into the `/data` folder of a session's *Data Volume*. This is shared, in read-only mode, within the *ServicePool* attached to the session. With the *Data Staging Workflow*, clients can always add data or, in case of updates, stage the same dataset multiple times. An important characteristic of SWIRRL is that it automatically keeps track of the history of the staging operations, detecting and communicating changes on specific data files. This has the effect of mitigating the impact of updates to the reproducibility of the scientific analyses, fostering, on

[⑧] https://esgf.llnl.gov/

the contrary, a consistent comparison of the results across versions. When the workflow downloads a file that already has a copy in the *Session*, the two files are compared to evaluate whether they carry the same content. In the case of no changes, the old copy is kept. Otherwise, a new version of the file is saved, keeping the old one in a dedicated sub-folder. Besides fostering reproducibility, this approach aims at improving the use of storage quotas, by avoiding to save duplicates of unchanged data. Eventually, version updates are traced in the provenance and communicated to the users[16]. The efficiency of the staging operations may be improved depending on the metadata offered by the data providers. For instance, the *ESGF Search*[17] service delivers a pre-calculated *checksum* as metadata attribute of each file of the federation, alongside its unique identifier. This can be used by the workflow to avoid downloading the data again in order to check for updates, thereby saving on computational costs. Data services vary depending on the research community. SWIRRL can host different implementations of the *Data Staging Workflow* depending on the characteristics of these services and on the requirements of the portal adopting the API. For instance, in *Climate4Impact*, SWIRRL supports remote data subsetting via a particular implementation that exploits remote processing end-points (WPS[18]). These are available at more locations of the ESGF and are interrogated by the workflow to obtain reduced data, moving the computational load at the data providers.

The aggregation of distributed data often brings other challenges associated with access rights. In the case of CORDEX data[19], we have approached this by implementing Single Sign On (SSO) between *Climate4Impact*, SWIRRL the new ESGF IdEA (Identity, Entitlement and Access management) [5], which supports the OpenID 2.0[20] protocol. SWIRRL's workflows use the OpenID delegation token to send a download request on behalf of the user who owns the workspace. The token is then checked by the IdEA system to grant access to the data, according to the user's rights. This protocol being widely adopted (e.g., also by EPOS) will facilitate us to apply this solution to other communities. Finally, all the staging workflows are implemented in CWL, taking advantage of the *scatter* instruction[21] to enable the parallelisation of each download.

5. AUTOMATING NOTEBOOK REPRODUCIBILITY

Pursuing reproducible research in Jupyter Notebook requires users to adopt general good practices [13] and learn how to use complementary systems [1]. We illustrate how SWIRRL assists users in this objective.

[16] https://dev.climate4impact.eu/c4i-frontend/helpSwirrl

[17] https://esgf.github.io/esg-search/ESGF_Search_RESTful_API.html

[18] https://www.ogc.org/standards/wps

[19] https://cordex.org/

[20] https://openid.net/specs/openid-authentication-2_0.html

[21] https://www.commonwl.org/user_guide/23-scatter-workflow/index.html

5.1 Updating and Restoring Notebook Libraries

When using Jupyter Notebook it is a common practice to update the underlying environment with new libraries or new versions of the already existing ones. SWIRRL detects such an update and records provenance of it, see Figure 3. This allows SWIRRL to keep track ofthe changes, from the creation of the *Service*, recording at each step the collection of installed libraries. Each library is described by its version and installation mode (i.e., via the *conda* or *pip* package managers). The information recorded in the provenance allows users to investigatechanges in the environment, especially when these affect the results. This can be performed interactively, thanks to the SWIRRL *Jupyter Extension* (Figure 4), where users can decide to rollback and restore to a previous state.

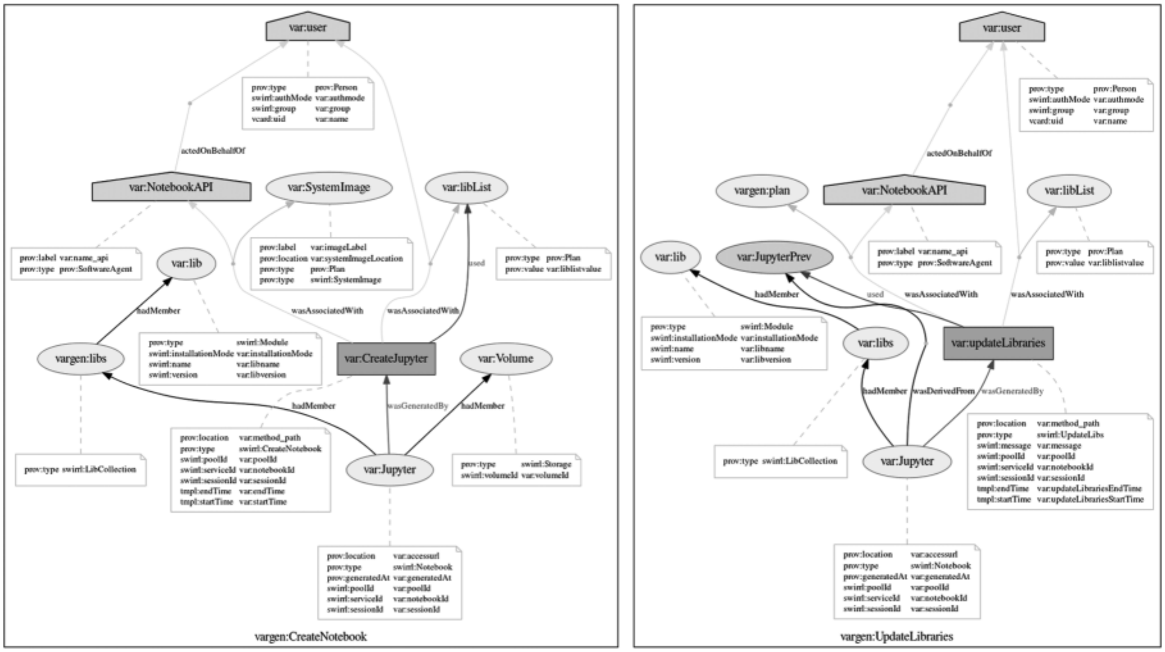
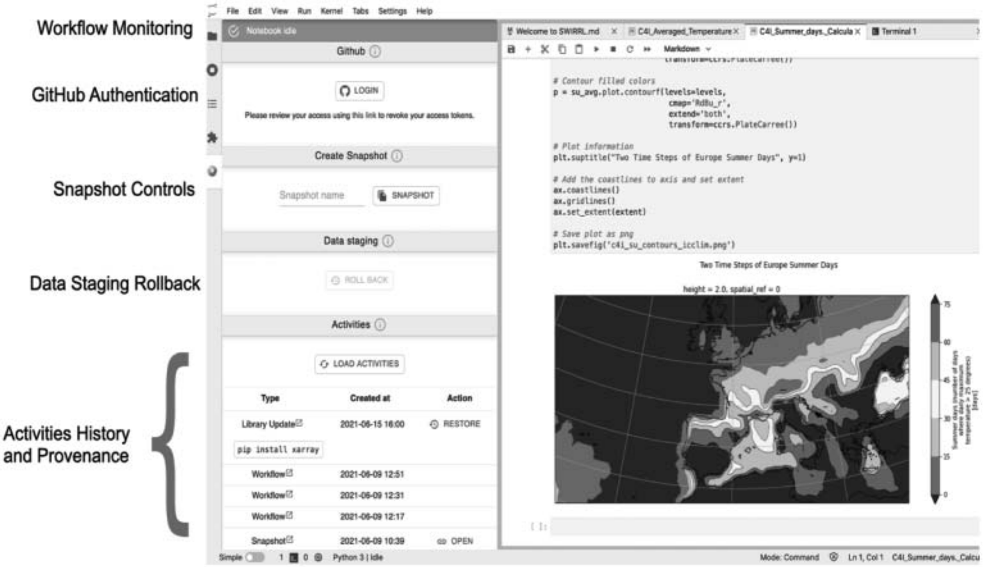

var:user
prov:type prov:Person
swirrl:authMode var:authmode
swirrl:group var:group
vcard:uid var:name
actedOnBehalfOf
var:NotebookAPI
var:SystemImage
var:libList
prov:label var:name_api
prov:type prov:SoftwareAgent
var:lib
prov:label var:imageLabel
prov:location var:systemImageLocation
prov:type prov:Plan
prov:type swirrl:SystemImage
used
prov:type prov:Plan
prov:value var:liblistvalue
hadMember
wasAssociatedWith
wasAssociatedWith
vargen:libs
prov:type swirrl:Module
swirrl:installationMode var:installationMode
swirrl:name var:libname
swirrl:version var:libversion
var:CreateJupyter
var:Volume
hadMember
wasGeneratedBy
hadMember
prov:type swirrl:LibCollection
prov:location var:method_path
prov:type swirrl:CreateNotebook
swirrl:poolId var:poolId
swirrl:serviceId var:notebookId
swirrl:sessionId var:sessionId
tmpl:endTime var:endTime
tmpl:startTime var:startTime
var:Jupyter
prov:type swirrl:Storage
swirrl:volumeId var:volumeId
prov:location var:accessurl
prov:type swirrl:Notebook
prov:generatedAt var:generatedAt
swirrl:poolId var:poolId
swirrl:serviceId var:notebookId
swirrl:sessionId var:sessionId
vargen:CreateNotebook
var:user
prov:type prov:Person
swirrl:authMode var:authmode
swirrl:group var:group
vcard:uid var:name
actedOnBehalfOf
vargen:plan
var:NotebookAPI
var:libList
var:lib
var:JupyterPrev
prov:label var:name_api
prov:type prov:SoftwareAgent
prov:type prov:Plan
prov:value var:liblistvalue
hadMember
used
wasAssociatedWith
wasAssociatedWith
prov:type swirrl:Module
swirrl:installationMode var:installationMode
swirrl:name var:libname
swirrl:version var:libversion
var:libs
var:updateLibraries
hadMember
wasDerivedFrom
wasGeneratedBy
prov:type swirrl:LibCollection
var:Jupyter
prov:location var:method_path
prov:type swirrl:UpdateLibs
swirrl:message var:message
swirrl:poolId var:poolId
swirrl:serviceId var:notebookId
swirrl:sessionId var:sessionId
tmpl:endTime var:updateLibrariesEndTime
tmpl:startTime var:updateLibrariesStartTime
prov:location var:accessurl
prov:type swirrl:Notebook
prov:generatedAt var:generatedAt
swirrl:poolId var:poolId
swirrl:serviceId var:notebookId
swirrl:sessionId var:sessionId
vargen:UpdateLibraries

Figure 3. Notebook service: CREATE and UPDATE PROV Templates. Both provenance templates record the presence of a particular version of a software library. This allows users to trace and recover the versions of the libraries installed across updates of the same Notebook environment, or to compare different environments, for instance, when these aim at repeating similar studies.



```
# Contour filled colors
p = su_avg.plot.contourf(levels=levels,
                         cmap='RdBu_r',
                         extend='both',
                         transform=ccrs.PlateCarree())

# Plot information
plt.suptitle("Two Time Steps of Europe Summer Days", y=1)

# Add the coastlines to axis and set extent
ax.coastlines()
ax.gridlines()
ax.set_extent(extent)

# Save plot as png
plt.savefig('c4i_su_contours_icclim.png')
```

Type	Created at	Action
Library Update `pip install xarray`	2021-06-15 16:00	RESTORE
Workflow	2021-06-09 12:51	
Workflow	2021-06-09 12:31	
Workflow	2021-06-09 12:17	
Snapshot	2021-06-09 10:39	OPEN

Figure 4. SWIRRL *JupyterLab* extension: interactive panel with monitoring and user controls to manage reproducibility and recovery actions. Users browse the activity history to read about changes in the underlying environment and trigger restoring actions, as well as access snapshots and workflows' provenance.

5.2 Notebook Snapshots

One of the benefits of using Jupyter is that the analysis methods are documented in the Notebook pages, which can be shared and interactively executed. However, to make sure the methods are reproducible, the organisation and sharing of the material, such as software dependencies, intermediate results and references to input data, require discipline from the author and manual work. SWIRRL reduces the burden on the researcher by offering a service that, upon request, automatically generates a reproducible snapshot of the current state of the Notebook. A snapshot includes details of the underlying environment, the most relevant artifacts produced for and by the analysis, and the URLs of the input data stored in the *Session*. This is pushed to GitHub in the form of a *Binder* repository, which can be shared and restored in any managed deployment of *My-Binder*[2]. Restoring directly to SWIRRL is also foreseen and currently under development. Users request the generation of a snapshot interactively via the SWIRRL *Jupyter Extension* (Figure 4). Here they can log into GitHub, to store the snapshots in their own account or use an institutional account associated with a particular instance of SWIRRL. The URLs of the repositories are always accessible via the *Jupyter Extension*. Snapshots can serve as a personal actionable record or for dissemination purposes. For instance, users can tag milestones within their progress that can be re-evaluated and reused in the future or produce educational material. The process is concluded with the generation of a provenance data, which makes sure that the role of each artifact involved in a snapshot is properly linked and described by rich metadata.

[2] http://mybinder.org

6. EPOS DATA PORTAL AND SWIRRL INTEROPERATION

The European Plate Observing System—EPOS, is a long-term plan to facilitate integrated use of data, data products, software and services from distributed research infrastructures in Europe in the Solid Earth Domain. Its innovation potential consists of the capability of integrating distributed heterogenous resources from European, National and institutional resource providers, in different thematic communities (e.g., Volcanology or Seismolgy), and making them available in one single environment: the EPOS Data portal[23]. Its functionalities enable users to find, pre-visualize and contextualize multidisciplinary datasets. In order to access advanced functionalities like processing, visualization and analysis, the system underpinning the portal (Integrated Core Services Central Hub—ICS-C) can interoperate with distributed e-infrastructures (Integrated Core Services Distributed—ICS-D) where datasets can be further studied, analysed and/or processed as part of a workflow. In this section we will discuss a ICS-D based on SWIRRL.

6.1 EPOS Interactive Workspaces

The EPOS Data portal is based on a user-friendly user interface which provides intuitive visualization methods and interaction modes that significantly facilitate the discovery and the access to the multidisciplinary datasets and services from the thematic communities. Its main functionalities include: (1) *Search*, by means of cross-domain search criteria; (2) *Browsing of results*; (3) *pre-visualization* of potential result candidates in order to verify whether these satisfy the user requirement; (4) *Refinement of results*, that can be further sub-set on the basis of domain specific search criteria; (5) *Data access*. The latter is accomplished via interactive workspaces. Here users store the results of their search, consisting of a set of URLs referencing to particular datasets, pre-defined workflows or other assets. They can make use of external e-Infrastructures built with the purpose of analysing or visualizing the data, or running workflows. The EPOS Data portal is undergoing the integration of SWIRRL in its workspaces to enable the users to accomplish the data research lifecycle, from data-discovery to analysis, including the comprehensive curation of their progress (Figure 5). We present a scientific use case in the field of seismology to discuss the practical use of SWIRRL in such context.

6.2 Use Case—Do Earthquakes Occur Only in the Crust or Also Below Moho?

The use case that demonstrates the integration of SWIRRL in the EPOS Data portal, has been proposed as an exercise in a workshop organised by the EPOS-Norway project[24]. It interests geophysicists and seismologists that use Python and Jupyter Notebook, in combination with *Enlighten-web* as a tool for visual analysis of geological and geophysical data, to answer the question "Do earthquakes occur only in the crust or also below Moho?"[25]. Its realisation encompasses three main steps: a) *search and contextualization phase*, where the user searches for the dataset of interest using the EPOS Data Portal functionalities; b) *saving the*

[23] https://www.ics-c.epos-eu.org/

[24] https://epos-no.uib.no

[25] https://earthquake.usgs.gov/learn/glossary/?term=Moho.

selected assets into the EPOS interactive workspace, c) *staging the selected assets* to a SWIRRL resource, d) *obtain combined access to data and interactive services*. While steps a) and b) are native functionalities of the EPOS Portal, the remaining steps are controlled via the SWIRRL API, with the additional support of provenance and reproducibility.

More specifically, users connect to the EPOS Data Portal in order to search the seismic events provided from the webservices of the Norwegian National Seismic Network (NNSN[26]), or from other data providers (e.g., EMSC European infrastructure for seismological products[27]), refining the search phase by setting appropriate values to the parameters used by the portal to query the different webservices. The data endpoints resulting from the search are then stored in the EPOS interactive workspace. Through the use of the SWIRRL API, the EPOS Data Portal associates the workspace with a *ServicePool* and a *Session*, providing user access to services such as *JupyterLab* and *Enlighten-web*. Upon request, the workspace generates a file containing the data URLs and passes it to the *Data Staging Workflow* for download. Once the data is staged, the researchers implement a Notebook to integrate the seismic data with particular information about the *Moho depth* [10], which can be manually uploaded. To do this, they update the Notebook with the Obspy[28] module and use this to create Pandas DataFrames[29] from the datasets in input. Thanks to the shared *Work Volume* between the services in the pool, users can access the DataFrames in *Enlighten-web*. Here, by co-visualising and slicing the two datasets interactively, they obtain visual evidence that the events occurred above the Moho (Figure 5). Finally, users can request to generate a reproducible snapshot of their Notebook. Thanks to the provenance recordings about the installation of Obspy in the Jupyter environment and about the *Data Staging* Workflow, SWIRRL will generate a coherent snapshot, thereby ensuring that both environment and data can be reproduced in the context of the same analysis.

[26] https://nnsn.geo.uib.no/nnsn/#/

[27] https://www.epos-eu.org/tcs/seismology/data-services/emsc-seismological-products-services

[28] https://docs.obspy.org/

[29] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

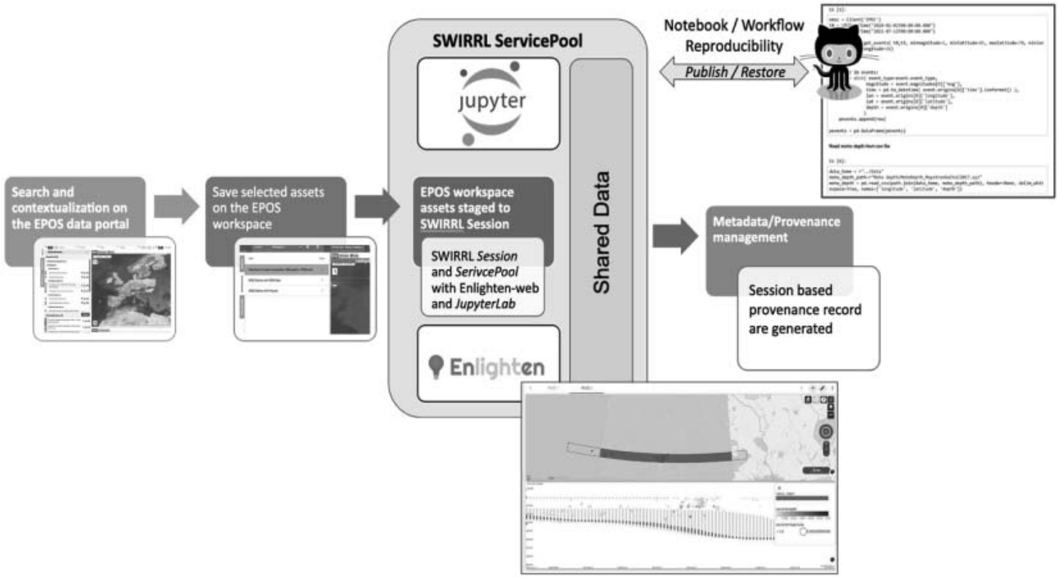

Search and contextualization on the EPOS data portal
Save selected assets on the EPOS workspace
SWIRRL ServicePool
jupyter
EPOS workspace assets staged to SWIRRL Session
SWIRRL Session and SerivcePool with Enlighten-web and JupyterLab
Enlighten
Shared Data
Notebook / Workflow Reproducibility
Publish / Restore
Metadata/Provenance management
Session based provenance record are generated

Figure 5. EPOS interactive-research workflow via SWIRRL. After having searched for data, users manipulate and visualise seismic and geological data within reproducible workspaces offering tools, such as *JupyterLab* and *Elinghten-web*.

7. CONCLUSIONS AND FUTURE WORK

SWIRRL hides the complexity of deploying user-facing services on common cloud platforms, allowing community portals to obtain data-driven workspaces. This is pursued by taking into account general reproducibility and interoperability challenges at level of the workflows and the environments that run the analysis tools. We discussed SWIRRL's particular adoption by community portals, where combining access to particular workflows with a flexible orchestrations of the workspaces enable researchers to (a) obtain data from federated community services, and (b) conduct custom investigations via different types of tools. At the time of the preparation of this paper, workflows are encoded within the API, which manages execution queues, deployment, monitoring and provenance recording. Future work will extend the API with a workflow registry [7], to improve and broaden the support of common operations.

We have illustrated how SWIRRL assists users in recovery and reproducibility actions of Jupyter Notebook. Thanks to the provenance recordings, updates to the software running in the notebooks and the data are linked to the the progress of the researcher's study, who can eventually generate metadata-rich milestones in the form of re-enactable snapshots. This is currently being extended to enable the reproducibility of visual analytic configurations in *Enlighten-web*.

Authorisation policies will be further developed. We will enable in SWIRRL the re-enactment of snapshots from private repositories, adding access-control mechanisms to enable access to protected input data and provenance by peers. We believe that our approach to the realisationof SWIRRL as a modern e-infrastructure service, contributes to address the challenges posed by the CWFR. In that respect, we apply separation of concerns between technical and research developers, pursuing the FAIR management of those research practices, where humans interact with a variety of systems to achieve their goals.

Outputs of SWIRRL are data products, notebook pages, snapshots and, as part of ongoing work, configuration profiles describing visual analytic setups. We envisage that each of these outputs could be packaged and delivered as FDOs. These will carry complete provenance traces describing the underlying environments, as well as the location of the raw data used for their generation. Such metadata can be extracted and encapsulated into the digital object's metadata layer, or referenced by linking to the provenance repository of SWIRRL, which builds on W3Cstandards. Moreover, the work of CWFR will help in identifying those operations that could betranslated into more general methods of the SWIRRL API. This would ease the integration of additional scientific tools, including the controls of common reproducibility actions, fostering thereby the adoption of our system by more research communities.

AUTHOR CONTRIBUTION

A. Spinuso (alessandro.spinuso@knmi.nl) and M. Veldhuizen (mats.veldhuizen@knmi.nl) contributed to the design and realisation of SWIRRL, including its integration in the *Climate4Impact* portal. D. Bailo (daniele.bailo@ingv.it) and V. Vinciarelli (valerio.vinciarelli@ingv.it) evaluated the conceptual and technical "fit for purpose" of the proposed technology in the context of the EPOS Data Portal and initiated its implementation in the EPOS workspaces. T. Langeland (tola@norceresearch.no) coordinated the support for *Enlighten-web* and provided the scientific use case used for the discussion. All authors contributed to writing and revising the manuscript.

ACKNOWLEDGEMENTS

The core development of SWIRRL has been conducted by the Observations and Data Technology Department of KNMI, with special contributions by Ian van Der Neut and Friedrich Striewski. This work is supported by the EU H2020 project ENVRIFair (No. 824068) and ISENES3 (No. 824084). We thank all software engineers and researchers involved in these projects for their support in adopting and testing the system, providing feedback for the progress of this research.

REFERENCES

[1] Beg, M., et al.: Using Jupyter for reproducible scientific workflows. Computing in Science Engineering 23(2), 36–46 (2021)

[2] Davidson, S., Freire, J.: Provenance and scientific workflows: Challenges and opportunities. In: Proceedings of the2008 ACM SIGMOD International Conference on Management of Data, pp. 1–6 (2008)

[3] De Smedt, K., Koureas, D., Wittenburg. P.: FAIR digital objects for science: From data pieces to actionable knowledge units. Publications 8(2), Article No. 21 (2020)

[4] Iturbide, M., et al.: Climate4r: An R-based open framework for reproducible climate data access and post-processing. Environmental Modelling & Software 111, 42–54 (2019)

[5] Kershaw, P., et al.: ESGF future architecture report. Technical report, Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States) (2020)

[6] Khan, F. Z., et al.: CWL Prov—interoperable retrospective provenance capture and its challenges. F1000Research 7 (2018). Available at: https://doi.org/10.7490/f1000research.1115721.1. Accessed 24 November 2021

[7] Klampanos, I., et al.: Dare: A reflective platform designed to enable agile data-driven research on the cloud. In: Proccedings of *eScience Workshop BC2DC*, No. 19473092 (2019)

[8] Knuth, D.E.: Literate programming. The Computer Journal 27(2), 97–111 (1984)

[9] Langeland, T., et al.: Epos-Norway portal. In: Geophysical Research Abstracts 21 (2019)

[10] Maystrenko, Y., et al.: Deep structure of the lofoten-vesterålen segment of the mid-norwegian continental margin and adjacent areas derived from 3-d density modeling. Journal of Geophysical Research: Solid Earth 122(2),1402–1433 (2017)

[11] Missier, P., et al.: D-prov: Extending the prov provenance model with workflow structure. In: Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance, pp. 1–9 (2013)

[12] Moreau, L., et al.: A templating system to generate provenance. IEEE Transactions on Software Engineering 44(2),103–121 (2017)

[13] Rule, A., et al.: Ten simple rules for reproducible research in Jupyter notebooks. arXiv preprint arXiv:1810.08055 (2018)

[14] Rule, A., Tabard, A., Hollan, J.D.: Exploration and explanation in computational notebooks. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, pp. 1–12 (2018)

[15] Spinuso, A., Atkinson, M., Magnoni. F.: Active provenance for data-intensive workflows: Engaging users and developers. In: The 15th International Conference on eScience (eScience), pp. 560–569 (2019)

[16] Ustyuzhanin, A., et al.: Everware toolkit. Supporting reproducible science and challenge-driven education. Journal of Physics: Conference Series 898, 072051 (2017)

AUTHOR BIOGRAPHY

Dr **Alessandro Spinuso** is a researcher at the R&D Observations and Data Technology division of the Royal Netherlands Meteorological Institute (KNMI). He earned his Ph.D. in Computer Science at the University of Edinburgh (UK) in 2017. At KNMI, he covers the roles of Researcher and Product Owner within an Agile R&D team developing Provenance-aware Data Analysis services. His main research interest is the management and the exploitation of provenance information in the context of user controlled computational environments providing notebooks and workflow systems for data-intensive analysis. He is involved in several EU initiatives (H2020, Copernicus), focusing on the development of e-science infrastructures for Earth Science research in Europe (EPOS, ENVRIFair, IS-ENES3, DARE, C3S). More recently, he is an invited expert to the IPCC TG-Data. A working group dedicated to the FAIR management of the data and methods that will be published in the next IPCC reports.
ORCID: 0000-0002-0077-8491

Mats Veldhuisen is a software developer for the Royal Netherlands Meteorological Institute (KNMI). Here he worked on the development of SWIRRL API, as well as other projects, in which there was always a focus on provenance. Before this he completed a double bachelor in Mathematics and Physics, followed by a Master’s degree in Computing Science at the university of Utrecht.
ORCID: 0000-0001-7894-9548

Dr **Daniele Bailo** received the Graduate degree in Computer Sciences and Engineering and a Ph.D. degree in Material Science. From 2005 to 2011 he has been collaborating with the ″Istituto di Struttura della Materia″ (ISM)—National Research Council (CNR), Rome, with the main task of designing, setting up and implementing Virtual X-Ray Spectrometry Laboratories for Web experiments, and carrying on research on plastic solar cells. From 2011 he worked at ″Istituto Nazionale di Geofisica e Vulcanologia″ (INGV) on the European Plate Observing System (EPOS) project as Research Infrastructure manager and designer, and IT developments coordinator. Since 2019 he has been Technical Officer of the EPOS-ERIC Executive and Coordination office, with the role of coordinating technical developments of the EPOS Integrated Core Services platform. His current research interest and topics of expertise include: e-infrastructure management and design; interoperability of systems; metadata standards for interoperability; Virtual Research Environments; turning FAIR into real technical implementation. He is author of several peer-reviewed scientific journal articles. He is also involved in several European projects and initiatives as principal investigator representing EPOS community, and also member of advisory board as ICT expert in system integration and GeoScience system design.
ORCID: 0000-0003-0695-4406

Valerio Vinciarelli is a software engineer for the EPOS ERIC Information Technology Unit. Since 2016, he has been involved in the design and realization of the technical architecture of the European Plate Observing Systems (EPOS—www.epos-eu.org). He received a bachelor's degree in Computer Science from the university of Rome, La Sapienza. He has five years of experience working in e-infrastructure management and design, interoperability of heterogeneous systems, Virtual Research Environments, and service-oriented architectures. He has been actively working in international initiatives and collaborations contributing to several EU projects.
ORCID: 0000-0001-5805-9337

Tor Langeland is a senior scientist affiliated with NORCE, a Norwegian research company. His main interests are within visualization and big data. Previous work includes development of systems for visualization of gas explosion data, illustrative visualizations of geological data and development of VR systems supporting cross disciplinary work processes in the oil & gas industry. Later projects have been focusing on development of Web based data portals for interactive visual analysis, e.g., in the context of the EPOS project.
ORCID: 0000-0003-2948-5761